AD-A236 524

ENTATION PAGE

Form Approved
OMB No. 0704-0188

1 AGENCY USE ONLY (Leave blank)	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	May 1991	Professional Paper

4 TITLE AND SUBTITLE
SUPERCONCURRENCY–A FORM OF DISTRIBUTED HETROGENEOUS SUPERCOMPUTING

5 FUNDING NUMBERS
PR: ECB2
WU: DN 300086
PE: 0602234N

6 AUTHOR(S)
R. F. Freund and D. S Conwell

7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152–5000

8 PERFORMING ORGANIZATION REPORT NUMBER

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
Block Programs
San Diego, CA 92152–5000

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11 SUPPLEMENTARY NOTES

12a DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b DISTRIBUTION CODE
A-1

13 ABSTRACT (Maximum 200 words)

Some of the complex issues of future supercomputing are discussed in the Richard Hill and John Gustafson papers published in the June 1990 issue of SUPERCOMPUTING REVIEW. Both papers contemplate the role of parallel processors in future operational environments. It is a variation of Gustafson's suggested approach, heterogeneous parallelism, that we will explore here, in particular Distributed Heterogeneous Supercomputing (DHS). DHS is the use of a heterogeneous suite of diverse processors, e.g., a mix of vector and parallel computers. DHS is not simply a LAN or a WAN because it aims to exploit the heterogeneous nature of the suite for reasons such as access to different data bases, access to remote special processors, or super-speed performance. Performance is the key to the Superconcurrency (Super-C) form of DHS, because Super-C tunes the selections of the different processors and optimally distributes the work primarily for maximum performance on the problem at hand (and only secondarily for load balancing).

Published in *Supercomputing Review*, Oct 1990.

91-01217

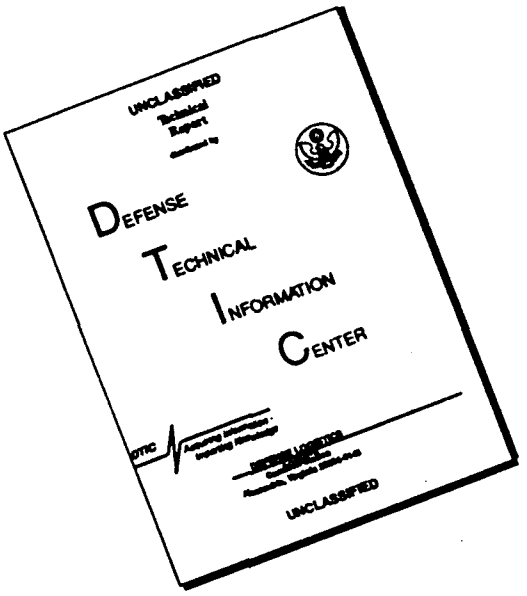
14 SUBJECT TERMS
massive parallel processing
high performance computing

15 NUMBER OF PAGES

16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

SUPERCONCURRENCY

A Form of Distributed Heterogeneous Supercomputing

BY RICHARD F. FREUND AND D. SUNNY CONWELL
NAVAL OCEAN SYSTEMS CENTER (NOSC)

Introduction

Some of the complex issues of future supercomputing are discussed in the Richard Hill and John Gustafson papers published in the June 1990 issue of *Supercomputing Review*. Both papers contemplate the role of parallel processors in future operational environments. We explore here a variation of Gustafson's suggested approach, heterogeneous parallelism, focusing in particular upon Distributed Heterogeneous Supercomputing (DHS). DHS is the use of a heterogeneous suite of diverse processors — e.g., a mix of vector and parallel computers.

DHS is not simply a LAN or a WAN because it aims to exploit the heterogeneous nature of the suite for reasons such as access to different data bases, access to remote special processors, or super-speed performance.

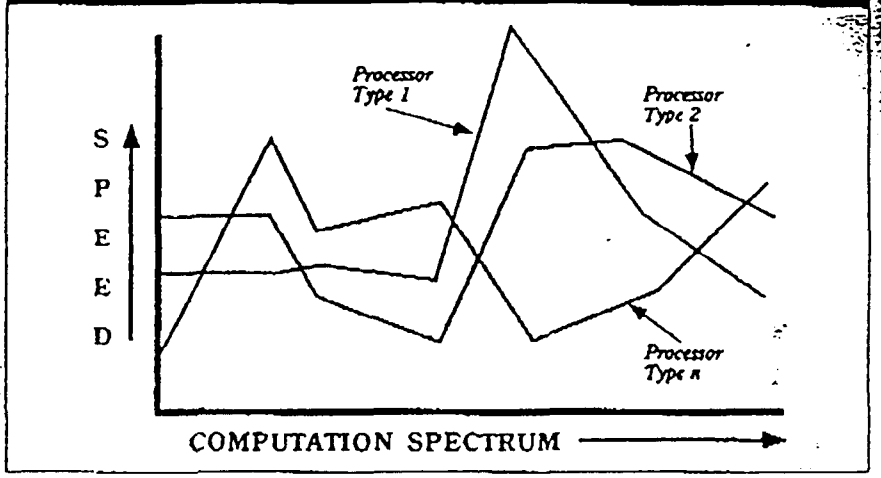
Performance is the key to the Superconcurrency (Super-C) form of DHS, because Super-C tunes the selection of the different processors and distributes the work optimally, primarily for maximum performance on the problem at hand and only secondarily for load balancing.

Types of Concurrency

There are a number of variant factors that enter into classifications of concurrent ("multiplicity") processing, e.g., memory organization (distributed, global, hierarchical, etc.) or processor interconnect scheme (bus, mesh, hypercube, etc.). Perhaps the most basic distinction is whether the processors execute the fine-grain parallelism of same instruction on multiple data (SIMD or vector) or the procedural parallelism of multiple instructions on multiple data (MIMD).

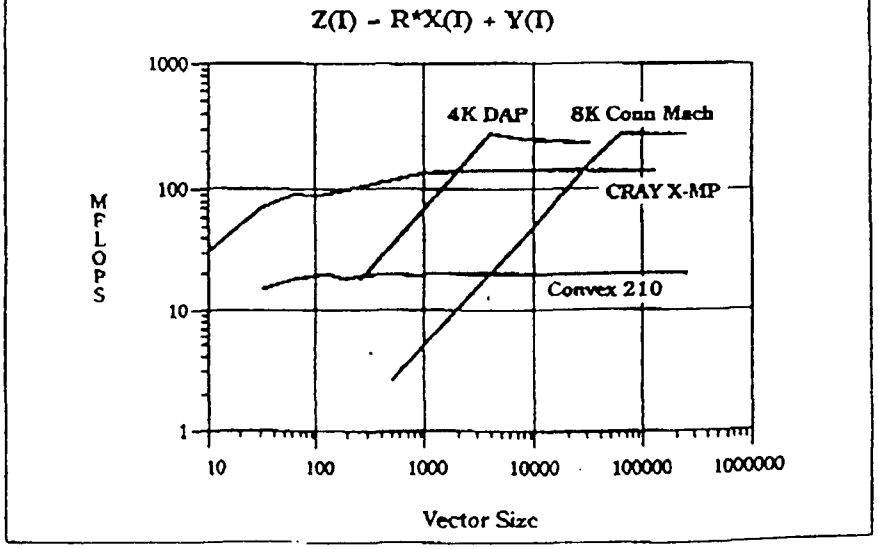
In any case it is only to be expected that quite different architectures would have different computation profiles, i.e., be relatively effective or ineffective on differing sections of the overall computation spectrum, as suggested heuristically in Figure 1 (where 'Computation Spectrum' is intended to suggest a wide range of computation tasks and parameter ranges).

Figure 1: Profiles of Processor Types



In addition, it is our experience that parameters and data lengths can affect the choice of architecture. Figure 2 demonstrates what we call cross-over points; it shows a CRAY X-MP, CONVEX 210, 8K Connection Machine (with co-processor), and 4K DAP on 32-bit SAXPY (using the compiler, not optimized subroutines).

Figure 2: Cross Over Points for SAXPY



SUPERCONCURRENCY, A FORM OF DISTRIBUTED HETEROGENEOUS SUPERCOMPUTING

Richard F. Freund and D. Sunny Conwell
Naval Ocean Systems Center (NOSC)

INTRODUCTION Some of the complex issues of future supercomputing are discussed in the Richard Hill and John Gustafson papers published in the June 1990 issue of SUPERCOMPUTING REVIEW. Both papers contemplate the role of parallel processors in future operational environments. It is a variation of Gustafson's suggested approach, heterogeneous parallelism, that we will explore here, in particular Distributed Heterogeneous Supercomputing (DHS). DHS is the use of a heterogeneous suite of diverse processors, e.g., a mix of vector and parallel computers. DHS is not simply a LAN or a WAN because it aims to exploit the heterogeneous nature of the suite for reasons such as access to different data bases, access to remote special processors, or super-speed performance. Performance is the key to the Superconcurrency (Super-C) form of DHS, because Super-C tunes the selection of the different processors and optimally distributes the work primarily for maximum performance on the problem at hand (and only secondarily for load balancing).

TYPES OF CONCURRENCY There are a number of variant factors that enter into classifications of concurrent ("multiplicity") processing, e.g., memory organization (distributed, global, hierarchical, etc.) or processor interconnect scheme (bus, mesh, hypercube, etc.). Perhaps the most basic distinction is whether the processors execute the fine-grain parallelism of same instruction on multiple data (SIMD or vector) or the procedural parallelism of multiple instructions on multiple data (MIMD). In any case it is only to be expected that quite different architectures would have different computation profiles, i.e., be relatively effective or ineffective on differing sections of the overall computation spectrum, as suggested heuristically in Figure 1 (where 'COMPUTATION SPECTRUM' is intended to suggest a wide range of computation tasks and parameter ranges).

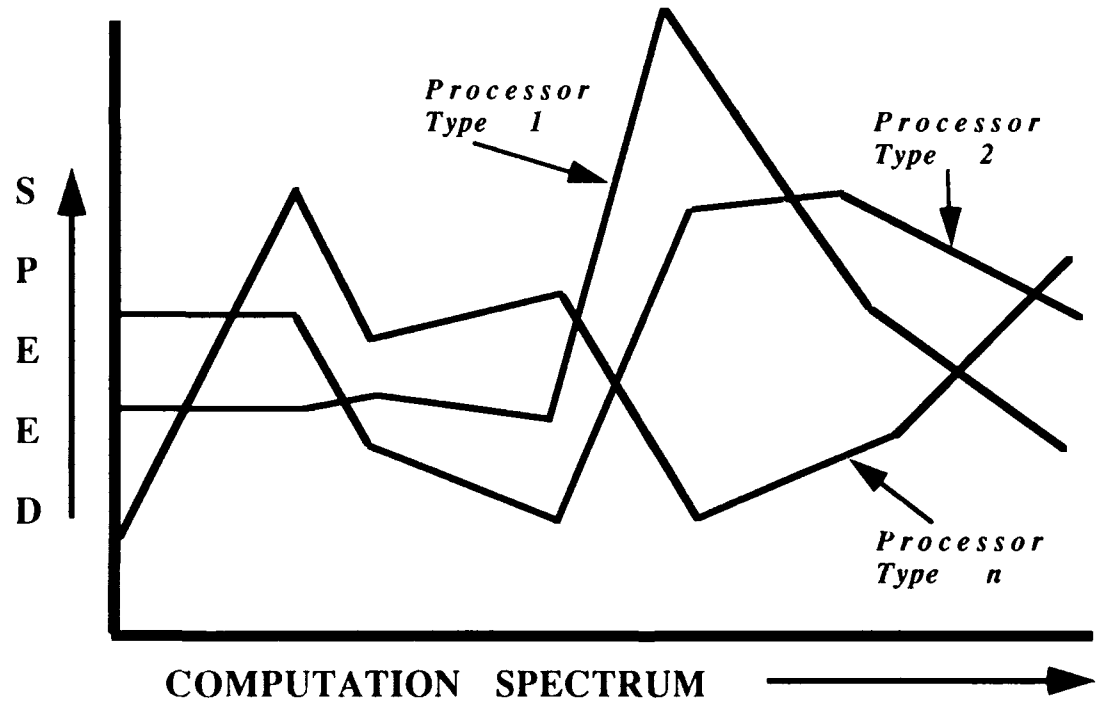


Figure 1. *Profiles of Processor Types*

In addition, it is our experience that parameters and data lengths can affect the choice of best architecture type. Figure 2 demonstrates what we call cross-over points; it shows a CRAY X-MP, CONVEX 210, 8K Connection Machine (with co-processor), and 4K DAP on 32-bit SAXPY (using the compiler, not optimized subroutines).

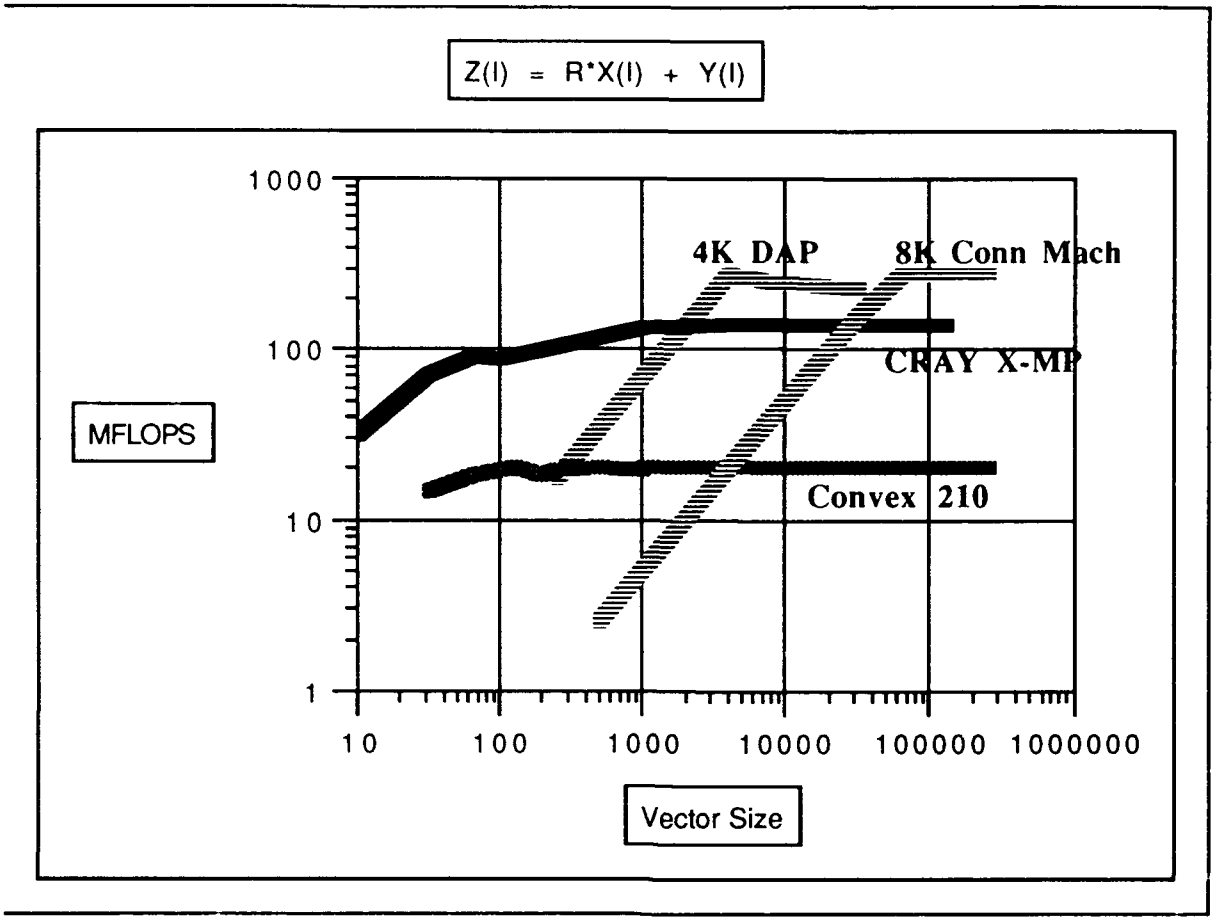


Figure 2. *Cross-over points for SAXPY*

AMDAHL'S LAW The need for DHS arises out of Amdahl's Law, or more precisely, a folk corollary of it:

> *Any single type of super-speed processor used on a heterogeneous code set often spends most of its time on the part it does poorest.*

To illustrate what we mean, suppose we had profiled a large code or set of codes on which we felt distinct, relatively loosely-coupled portions (see "DINS" below) would best be done on quite different architectures, e.g., vector, SIMD, MIMD, special purpose, and data-flow. Let us choose any single processor type, say a CRAY, which is primarily a vector engine. Using the CRAY we might drive the vector portion of the code down to virtually no time, but we would still be left with the non-vector portions (sometimes mistakenly simply called "scalar"). A several-processor CRAY might make modest gains on the MIMD portions and because of its relatively fast scalar processor, it might make some time reductions on the rest of the code. However it would not do nearly as well on these other, non-vector portions as would machines (much less costly than a CRAY) designed for those types of computations. We would be left with the CRAY spending most of its time on the code portions where it achieves only modest improvements. Of course we would obtain similar results if we chose any single processor type which would do well on its kind of code and only fair on other code types. We believe that a more sensible approach is to build a team of processor types that match the computation requirements, as illustrated in Figure 3.

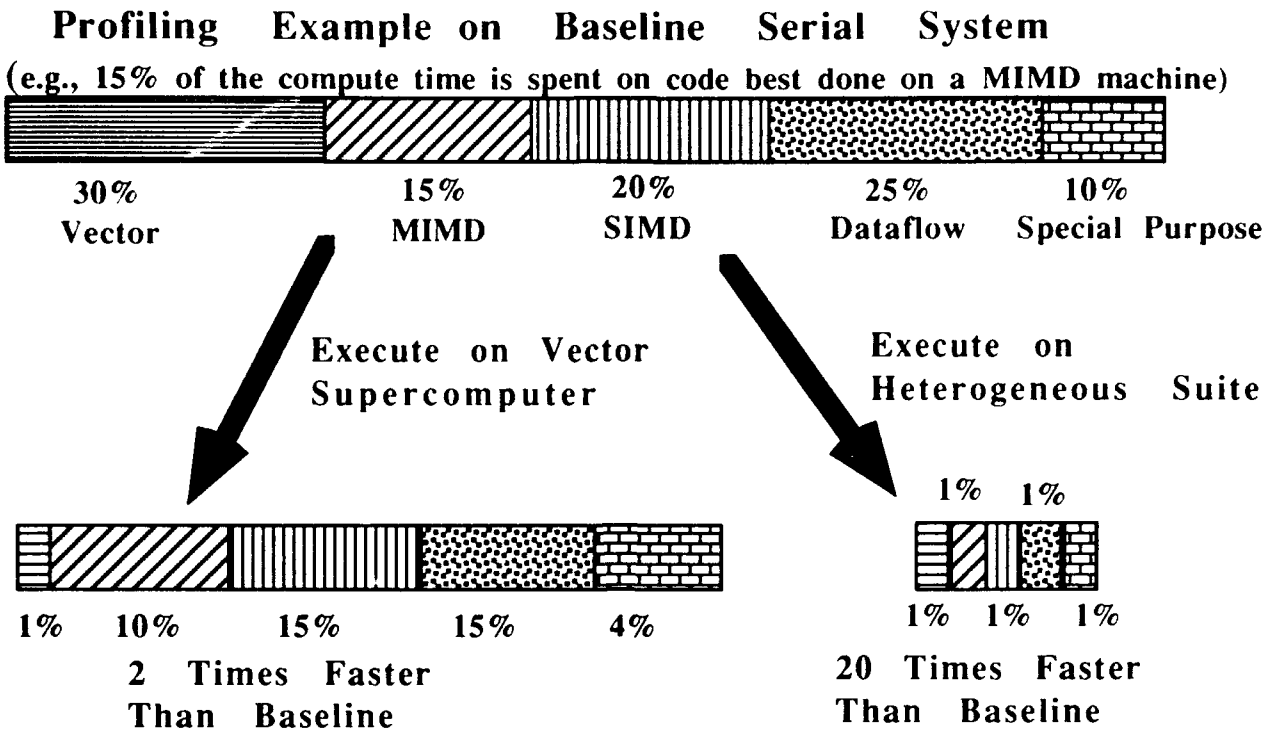


Figure 3. *Code Profiling and Machine Matching*

RESPONSES This effects of Amdahl's Law have been considered before and there are a set of traditional responses.

i. One response is to change the relative percentages of the code types. Suppose we wish to compute with only a vector machine and 35% of the execution time of our code (on some baseline serial system) is naturally vectorizable. Often one can work on changing the algorithm and code to increase this percentage. Typically, however, this approach achieves only modest benefits, say an increase to 50% vectorizable, and often takes a great deal of programmer effort.

ii. Another response has been single processor augmentation, e.g., bringing in a special second computer, such as an array processor, to handle suitable portions of the code. Typically this approach still leaves significant code portions that are not optimal matches to either the main processor or the special purpose machine (and therefore dominant in the overall timings).

iii. Our belief is that a more natural response is to use a tuned suite of heterogeneous processors. Thsi approach attempts to cover all the main types of computation required and to orchestrate effectively the use of the different processors. The potential advantages of this are clear; we optimize the match of all the different portions of code to processor types (wrt compute time) and potentially achieve much higher speeds than the use of any single supercomputer, however powerful.

TRENDS We perceive (Figure 4) four main trends in the development of supercomputing.

i. Device technology -- Advances in basic component technology, primarily density, size, and speed.

ii. Pipelining and vectorization -- Techniques for both scalar and vector hardware to compute at the (asymptotic) rate of one result per clock cycle.

iii. Homogeneous parallelism -- Use of one basic type of parallel design to solve a problem set.

iv. Distributed Heterogeneous Supercomputing -- The combined and orchestrated use of different vector and parallel processors to solve an application set with diverse computation requirements.

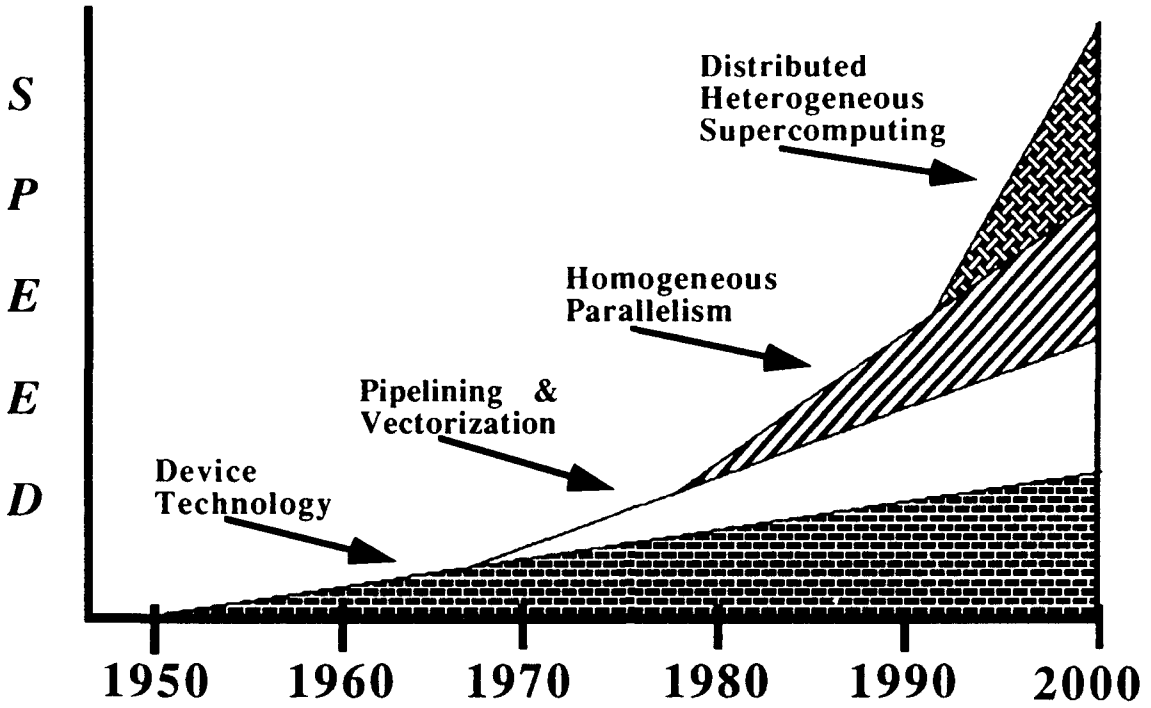


Figure 4. *Supercomputing Trends*

FORMS We see three distinct and potentially complementary forms of Distributed Heterogeneous Supercomputing in the near future.
i. Global DHS -- Heterogeneous processing done over a large geographical area, e.g., the Concurrent Supercomputing Project at JPL/CalTech (Figure 5A). This approach can be used not only to optimize computation speed, but also to maximize the use of remote data bases or display/interface capabilities.
ii. Site DHS -- Mid-sized DHS at one site. This form is currently being implemented at NOSC (Figure 5B).
iii. Micro DHS -- DHS in the small, in which all the diverse processors take the form of different processing functionality in one physical box, e.g., the Purdue mixed-mode PASM [1]. We anticipate seeing new ventures in this arena in the next few months, e.g., an Encore-DAP, with the conscious aim of developing self-contained, tightly-integrated, and tailorable superconcurrent processing (Figure 5C).

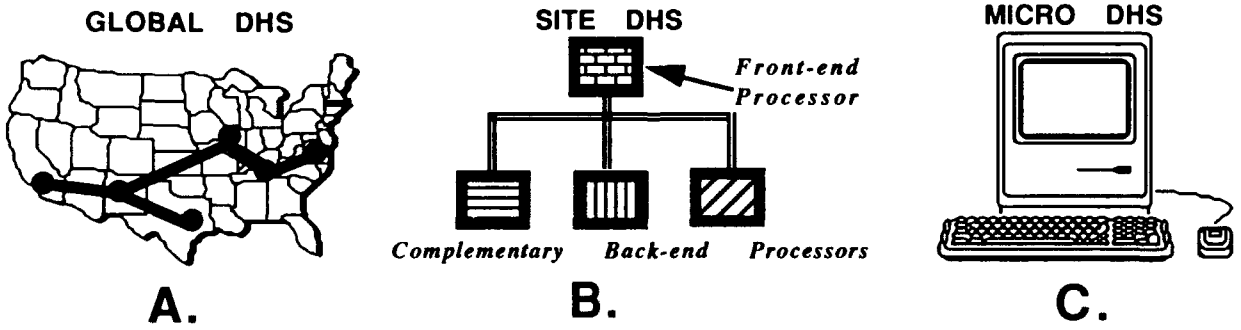


Figure 5. *Forms of DHS*

SUPERCONCURRENCY Superconcurrency is a form of DHS emphasizing performance gains resulting from optimally-configured vector and parallel mini-supers. The Superconcurrency Research Team at NOSC is evaluating this technology for Navy Command and Control problems as described in [2]. Superconcurrency is a general technique for matching and managing optimally configured suites of super-speed processors. In particular the reference demonstrates a general method (actually a mathematical program - eq 1. below) for choosing the most powerful suite of heterogeneous parallel and vector supercomputers for a given problem set, subject to a

fixed constraint, such as cost. The dual problem could find a minimal cost configuration for a fixed speed requirement.

$$\text{MINIMIZE } \mathcal{T} = \sum_{j=1}^{N} \min_{1 \le i \le M} \left(\frac{t_{i,j}}{v_i} \right)$$
$$\text{such that } \sum_{i=1}^{M} v_i \, c_i \le \mathcal{C} \qquad (1)$$

where N = # different code types, M = # different processor types, v_k = total # processors of type k, c_k = cost of processor type k, $t_{l,k}$ = time for processor l on code type k, $\mathcal{T}$ is the total time (function to be minimized), and $\mathcal{C}$ is the overall cost constraint. This approach, called the Optimal Selection Theory is mathematically dependent on new methodologies of code profiling and analytical benchmarking, as suggested by Figures 1 & 3 above.

DINS OR DYNAMIC OPTIMIZATION One of the most active current research areas of the NOSC Superconcurrency Research Team has been the development of the Distributed Intelligent Network System (DINS) concept. DINS will be a reasoning system, built upon an existing distributed O/S, that uses information from Code Profiling, Analytical Benchmarking, and network bandwidth to optimally manage a network of heterogeneous, high-performance, concurrent processors and assigns portions of code to appropriate processors. Superconcurrent implementations will work at the lowest level granularity compatible with the bandwidths available at any given site and the degree of coupling required by the various code modules. Put another way, equation 1 above will actually use $t'_{i,j}$ where the t' reflect not only the actual compute time for processor i on code type j, but the required interprocessor communication time as well:

$$\text{MINIMIZE } \mathcal{T} = \sum_{j=1}^{N} \min_{1 \le i \le M} \left(\frac{t'_{i,j}}{v_i} \right) \qquad (2)$$

In a general sense, this approach is similar to current research in load balancing and priority assignment. However the information sources will be the Profiling, Benchmarking, and bandwidth factors with the primary aim of optimal matching of code portions to processors rather than (the secondary) factors of load balancing and priority assignment. Since DINS will reason about processors actually available to it, we have the potential to achieve configuration control at different sites even though there may be a different superconcurrent suite at each. Similarly DINS will continue to function and assign a second best processor if a first choice is unavailable or down. Thus DINS is robust and survivable. Likewise it is compatible with evolutionary development; when a new processor is introduced because of changing technology, the old benchmarking data can be replaced with the new. The features of robustness, configuration control, survivability, tailorability, and evolutionary development are essential for many Navy problems. We call DINS dynamic optimization since it will dynamically task, in an optimal way, the backend suite of heterogeneous, superconcurrent processors that are chosen by the Optimal Selection Theory.

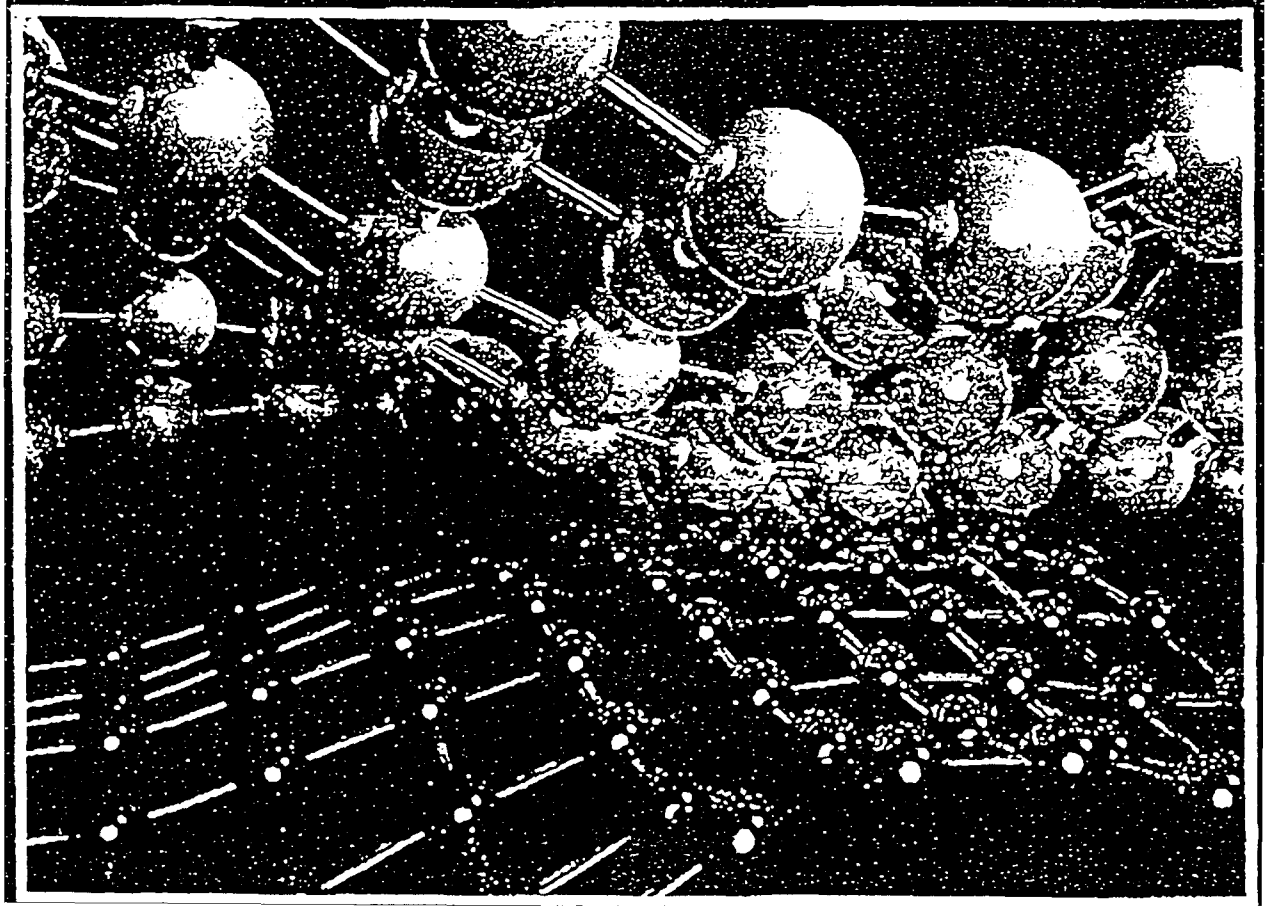
ACKNOWLEDGEMENTS This research is supported by the Office of Naval Technology and the Naval Ocean Systems Center.

REFERENCES

[1] Howard Jay Siegel, Thomas Schwederski, James T. Kuehn, and Nathaniel J. Davis IV, *An Overview of the PASM Parallel Processing System*, in Computer Architecture, edited by D. D. Gajski, V. M. Milutinovic, H. J. Siegel, and B. P. Furht, IEEE Computer Society Press, Washington, D.C., pp. 387-407, 1987.

[2] R. F. FREUND, *Superconcurrent Processing, A Dynamic Approach to Heterogeneous Parallelism*, Proceedings of the Parallel/Distributed Computing Networks Seminar, 24 February 1990, San Diego Section, IEEE.

Kodak Takes Flight from NCSA
Super Images from Science and Music



ndahl's Law

The need for DHS arises out of dahl's Law, or more precisely, a corollary of it:

Any single type of super-speed processor used on a heterogeneous code set often spends most of its time on the part it does poorest.

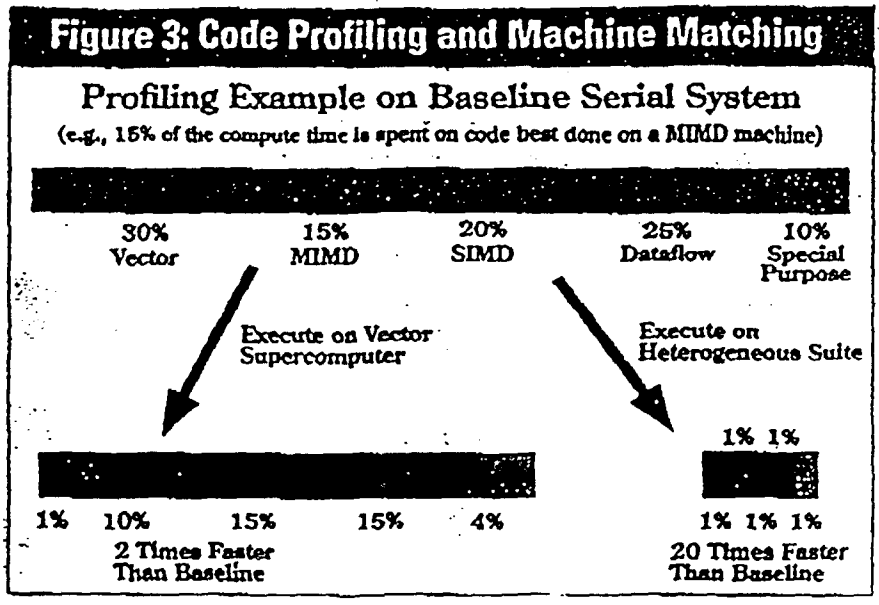
To illustrate, suppose we had pro- l a large code or set of codes vhich we felt distinct, relatively ely coupled portions (see "DINS" ow) would best be done on quite erent architectures, e.g., vector, D, MIMD, special purpose, and flow. Let us choose any single essor type, say a Cray, which is arily a vector engine. Using the we might drive the vector por- of the code down to virtually no , but we would still be left with the vector portions (sometimes mis- nly simply called "scalar"). A sev- processor Cray might make mod- ains on the MIMD portions. Because of its relatively fast sca- rocessor, the Cray might also some time reductions on the rest e code. However, it would not do ly as well on these other, non- r portions as would machines h less costly than a Cray) design- r those types of computations. We would be left with the Cray ding most of its time on the code ons where it achieves only mod- mprovements. Of course we l obtain similar results if we any *single* processor type that l do well on its kind of code and fair on other code types. We e that a more sensible approach uild a team of processor types atch the computation require- , as illustrated in Figure 3.

ponses

his effect of Amdahl's Law has considered before and there is f traditional responses.

ne response is to change the lative percentages of the code pes. Suppose we wish to com- te with only a vector machine, er speeds than the use of any single supercomputer, however powerful.

Figure 3: Code Profiling and Machine Matching



and 35 percent of the execution time of our code (on some baseline serial system) is naturally vectorizable. Often one can work on changing the algorithm and code to increase this percentage. Typically, however, this approach achieves only modest benefits, say an increase to 50 percent vectorizable, and often takes a great deal of programmer effort.

2) Another response has been single-processor augmentation, e.g., bringing in a special second computer, such as an array processor, to handle suitable portions of the code. Typically this approach still leaves significant code portions that are not optimal matches to either the main processor or the special-purpose machine (and therefore dominant in the overall timings).
3) Our belief is that a more natural response is to use a tuned suite of heterogeneous processors. This approach attempts to cover all the main types of computation required and to orchestrate effectively the use of the different processors. The potential advantages are clear: We optimize the match of all the different portions of code to processor types and potentially achieve much higher speeds than the use of any

Trends

We perceive four main trends in the development of supercomputing, shown in Figure 4.

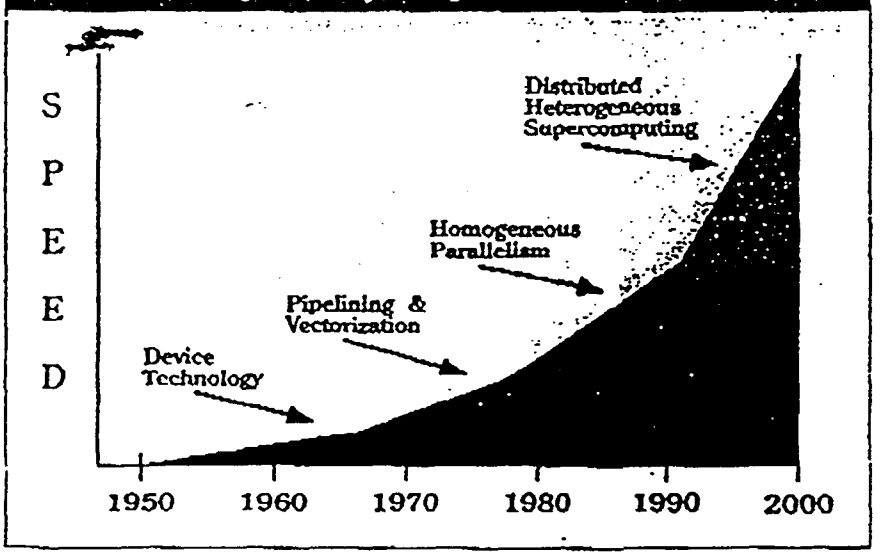
1) Device technology — Advances in basic component technology, primarily density, size and speed.
2) Pipelining and vectorization — Techniques for both scalar and vector hardware to compute at the (asymptotic) rate of one result per clock cycle.
3) Homogeneous parallelism — Use of one basic type of parallel design to solve a problem set.
4) Distributed Heterogeneous Supercomputing — The combined and orchestrated use of different vector and parallel processors to solve an application set with diverse computation requirements.

Forms

We see three distinct and potentially complementary forms of Distributed Heterogeneous Supercomputing in the near future.

1) Global DHS — Heterogeneous processing done over a large geographical area, e.g., the Concurrent Supercomputing

Figure 4: Supercomputing Trends



JPL/CalTech (Figure 5A). This approach can be used not only to optimize computation speed, but also to maximize the use of remote databases or display/interface capabilities.

2) Site DHS — Mid-sized DHS at one site. This form is currently being implemented at NOSC (Figure 5B).
3) Micro DHS — DHS in the small, in which all the diverse processors take the form of different processing functionality in one physical box, e.g., the Purdue mixed-mode PASM [1]. We anticipate seeing new ventures in this arena in the next few months, e.g., an Encore-DAP, with the conscious aim of developing self-contained, tightly-integrated, and tailorable superconcurrent processing (Figure 5C).

Superconcurrency

Superconcurrency is a form of DHS emphasizing performance gains resulting from optimally-configured vector and parallel mini-supers. The Superconcurrency Research Team at NOSC is evaluating this technology for Navy Command and Control problems as described in [2].

Superconcurrency is a general technique for matching and managing optimally configured suites of super-speed processors. In particular the reference demonstrates a general method (actually a mathematical program — Equation 1, below) for choosing the most powerful suite of heterogeneous parallel and vector supercomputers for a given problem set, subject to a fixed constraint, such as cost. The dual problem could find a minimal cost configuration for a fixed speed requirement.

$$\text{MINIMIZE } \mathcal{T} = \sum_{j=1}^{N} \min_{1 \le i \le M} \left(\frac{t_{i,j}}{v_i} \right)$$

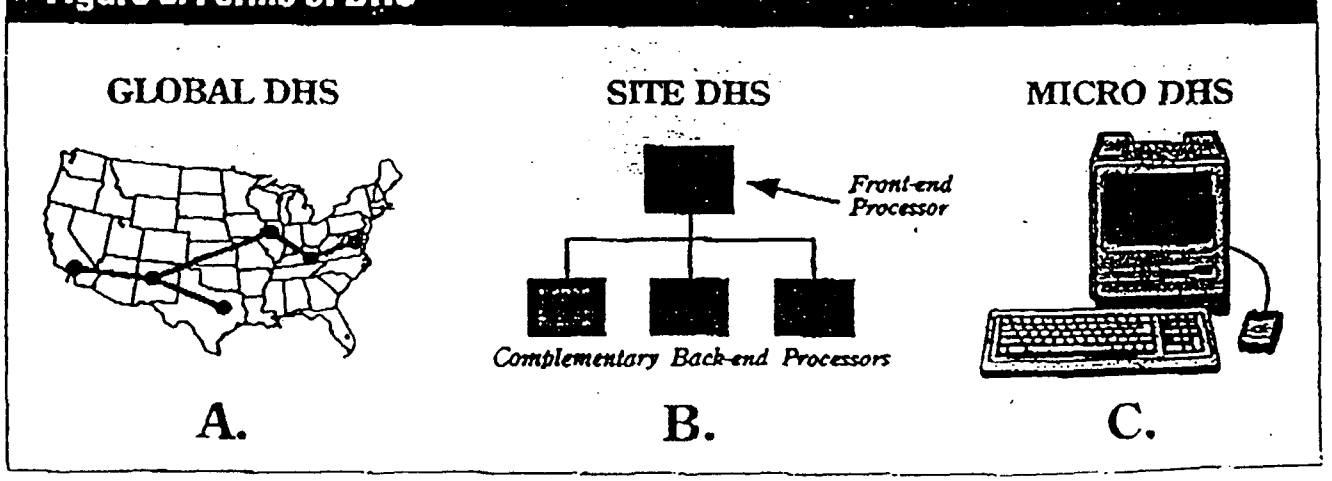
$$\text{such that } \sum_{i=1}^{M} v_i c_i \le C \qquad (1)$$

where N = # different code types, M = # different processor types, n_k = total # processors of type k, c_k = cost of processor type k; $t_{l,k}$ = time for processor l on code type k, $\mathcal{T}$ is the total time (function to be minimized), and C is the overall cost constraint. This approach, called the Optimal Selection Theory is mathematically dependent on new methodologies of code profiling and analytical benchmarking, as suggested by Figures 1 & 3 above.

DINS or Dynamic Optimization

One of the most active current research areas of the NOSC Superconcurrency Research Team has been the development of the Distributed Intelligent Network System (DINS)

Figure 5: Forms of DHS



oncept. DINS will be a reasoning stem, built upon an existing distributed O/S, that uses information from ode Profiling, Analytical Benchmarkg, and network bandwidth to optimalmanage a network of heterogeneous, igh-performance, concurrent proccssors. It assigns portions of code to ppropriate processors.

Superconcurrent implementations ill work at the lowest level of granurity compatible with the bandwidths vailable at any given site and the deree of coupling required by the varius code modules. Put another way, quation 1 above will actually use $t^i_{i,j}$ here the t^i reflects not only the actal compute time for processor i on ode type j, but the required interprocssor communication time as well:

$$\text{INIMIZE } \mathcal{T} = \sum_{j=1}^{N} \min_{1 \le i \le M} \left(\frac{t^i_{i,j}}{v_i} \right)$$

2)

In a general sense, this approach similar to current research in load alancing and priority assignment. owever, the information sources will be the profiling, benchmarking and bandwidth factors with the primary aim of optimal matching of code portions to processors rather than (the secondary) factors of load balancing and priority assignment.

Since DINS will reason about processors actually available to it, we have the potential to achieve configuration control at different sites even though there may be a different superconcurrent suite at each. Similarly, DINS will continue to function and assign a second-best processor if a first choice is unavailable or down.

Thus, DINS is robust and survivable. It is in addition compatible with evolutionary development. When a new processor is introduced because of changing technology, the old benchmarking data can be replaced with the new. The features of robustness, configuration control, survivability, tailorability, and evolutionary development are essential for many Navy problems. We call DINS dynamic optimization since it will dynamically task, in an optimal way, the back-end suite of heterogeneous, superconcurrent processors that are chosen by the Optimal Selection Theory.

Acknowledgments

This research is sur orted by the Office of Naval Technology and the Naval Ocean Systems Center.

References

[1] Howard Jay Siegel, Thomas Schwederski, James T. Kuehn, and Nathaniel J. Davis IV, *An Overview of the PASM Parallel Processing System, in Computer Architecture*, edited by D. D. Gajski, V. M. Milutinovic, H. J. Siegel, and B.P. Furht, IEEE Computer Society Press, Washington, D.C., pp. 387-407, 1987.

[2] R. F. Freund, *Superconcurrent Processing, A Dynamic Approach to Heterogeneous Parallelism*, Proceedings of the Parallel/Distributed Computing Networks Seminar, 24 February 1990, San Diego Section, IEEE.